Reinforcement Learning: An Introduction

Solutions: Chapter 2

Mrinank Sharma

March 27, 2020

Exercise 2.1 ϵ-Greedy Action Selection

$$\Pr[\text{greedy action selected}] = \epsilon \cdot \frac{1}{2} + (1-\epsilon) \tag{1}$$

$$= \frac{3}{4} \tag{2}$$

Exercise 2.2 Bandit Example

The exploratory action could have occured on any of the timesteps (and the behaviour could have coincidently given us the sequence of actions). However, the action **must** have occured on timesteps 2 and 5 since the greedy action was not selected.

Exercise 2.3 Infinite Horizon Performance

In the long run, assume that the sample averages have converged to the correct values or close enough for the exploratory cases. There is no guarantee of this happenign for the purely greedy strategy. Then, the probability of selecting the right action is 0.99 and 0.9 for $\epsilon = 0.01$ and $\epsilon = 0.1$ respectively and from the graph, about $\frac{1}{3}$ for the purely greedy example.

The greedy algorithm seems to average about 1 reward on each timestep (I'm sure there are mathematical reasons for this). So, in the long run, it would have cumulative reward T. The *expected value* of a purely exploratory action is 0, since the arm means are drawn from a $\mathcal{N}(0,1)$ distribution. As a result, in the long run:

$$\text{expected cumulative reward}_{\epsilon=0.01} = 0.99 \cdot 1.55 \cdot T \tag{3}$$

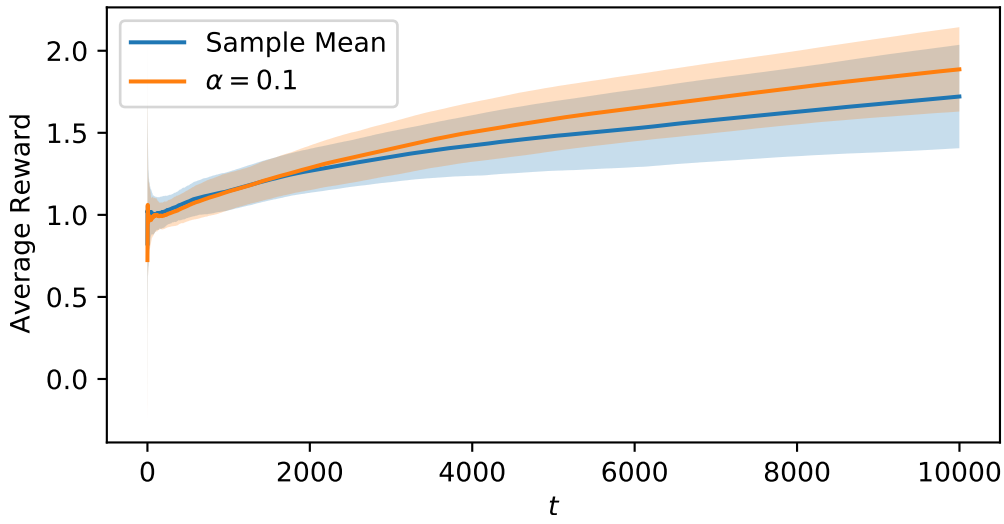
$$\text{expected cumulative reward}_{\epsilon=0.1} = 0.9 \cdot 1.55 \cdot T \tag{4}$$

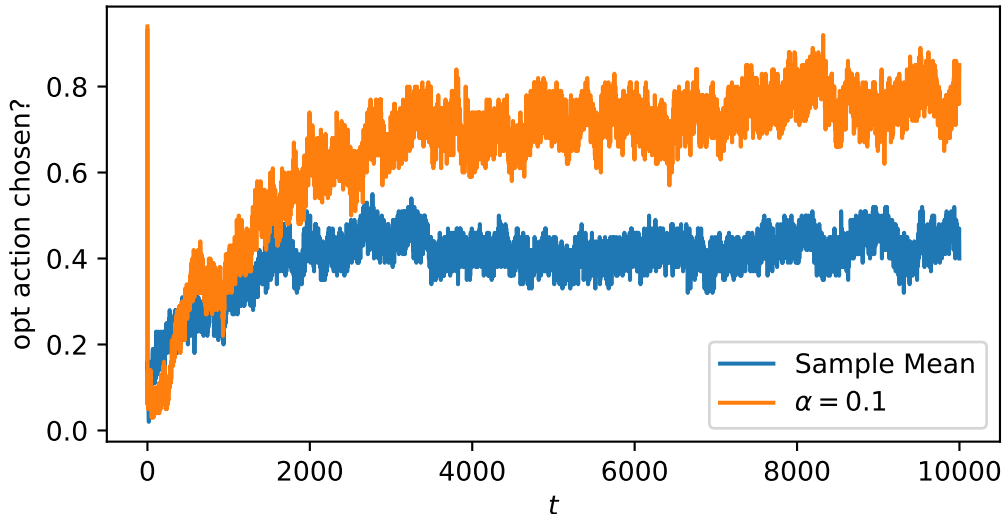
Exercise 2.4 Changing Step Sizes

Note: might have got the indices slightly confused here!

$$\begin{aligned} Q_{n+1} &= Q_n + \alpha_n \left[R_n - Q_n\right] \\ &= \alpha_n R_n + (1-\alpha_n) Q_n \\ &= \alpha_n R_n + (1-\alpha_n)\alpha_{n-1} R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1}) Q_{n-1} \\ &\vdots \\ &= \sum_{i=0}^{n-1} \alpha_{n-i} R_{n-i} \prod_{j=0}^{i-1} (1-\alpha_{n-j}) + Q_1 \prod_{i=0}^{n-1} (1-\alpha_{n-1}) \end{aligned} \tag{5}$$

Exercise 2.5 Programming





As expected, the sample mean method does not work so well. I expect it to get worse as more time goes on, but it still seems to do reasonably well.

Exercise 2.6 Mysterious Spikes

The spikes only occur for the optimistic, greedy method even though we've averaged across 2000 different bandit problems.

The algorithm has wildly too high initial values. This effectively means that for the first ten steps, the algorithm will explore every single option. After the 10th step, the algorithm will greedily pick the best option. This is most likely to be the true best option. However, eventually it will pick this option enough times that it no longer looks like the true best option (since the other estimates values are all too high). It will then pick those options until their value drops, and this keeps repeating.

This sort of behaviour won't get smoother as we randomise more. We are always dealings with updates from small numbers of rewards with poor initial values and as a result, the estimated value for each of the arms will be changing most at the start, meaning that there will be oscillations in the optimal action selected.

I'm not particularly convinced by this answer.

Exercise 2.7: Unbiased Constant Step Size Trick

Since $\bar{o}_0 = 0$, $\bar{o}_1 = \alpha$ so it is clear that $Q_2 = R_1$. Therefore, there is no initial bias.

We need to show that the remaining terms form an weighted average. Let's go back to our previous formula. Doing some relabeling, we can write Q_{n+1} as:

$$Q_{n+1} = \sum_{i=1}^{n} R_i \beta_i \prod_{j=i+1}^{n} (1 - \beta_j). \tag{6}$$

We want to show that the coefficients form a weighted average. It is easy to see that:

$$\beta_i = \frac{1}{\sum_{j=0}^{i-1}(1-\alpha)^j}. \tag{7}$$

$\beta_i > 0 \ \forall i$ and $\beta_i > \beta_{i+1}$. We want to show that:

$$\sum_{i=1}^{n} \beta_i \prod_{j=i+1}^{n} (1-\beta_j) = 1. \tag{8}$$

We now show that the coefficients sum to 1 by proof by induction. Consider the case $n = 1$. Clearly the coefficients sum to 1, since $\beta_1 = 1$. Now, assume that the above holds for $n = n^*$. Then,

$$\sum_{i=1}^{n^*+1} \beta_i \prod_{j=i+1}^{n^*+1} (1-\beta_j) = \beta_{n^*+1} + (1-\beta_{n^*+1}) \underbrace{\sum_{i=1}^{n^*} \beta_i \prod_{j=i+1}^{n^*} (1-\beta_j)}_{1} = 1 \tag{9}$$

If true for n, true for $n+1$. Since true for $n = 1$, true for all positive integers. Thus we have a exponentially weighted average with no initial bias.

Exercise 2.8: UCB Spikes

The algorithm will explore every single action first because all of the arm counts are zero. On the 11th step, it will pick what seems to be the best option **across all of the games**, giving the spike in most cases (depending on the exact draws, not always the best value), but then the uncertainty reward will go down, leading to other arms being picked.

If $c = 1$ is used, the spike is less prominent as the reward for uncertainty is smaller. The algorithm requires more draws from the best arm to sample other arms, giving a smoother curve. The larger the value of c, the more quickly the estimates are likely to be changing (due to large changes in the uncertainty), so the noisier the curves.

Exercise 2.9: Softmax

Suppose $a \in \{0, 1\}$. Then:

$$\begin{aligned} \Pr(A_t = 0) &= \frac{\exp H_t(0)}{\exp(H_t(0)) + \exp(H_t(1))} \\ &= \frac{1}{1 + \exp(-(H_t(0) - H_t(1)))}, \end{aligned} \tag{10}$$

Which looks just like the sigmoid function:

$$\sigma(z) = \frac{1}{1 + \exp(-z)}, \tag{11}$$

with $z = H_t(0) - H_t(1)$

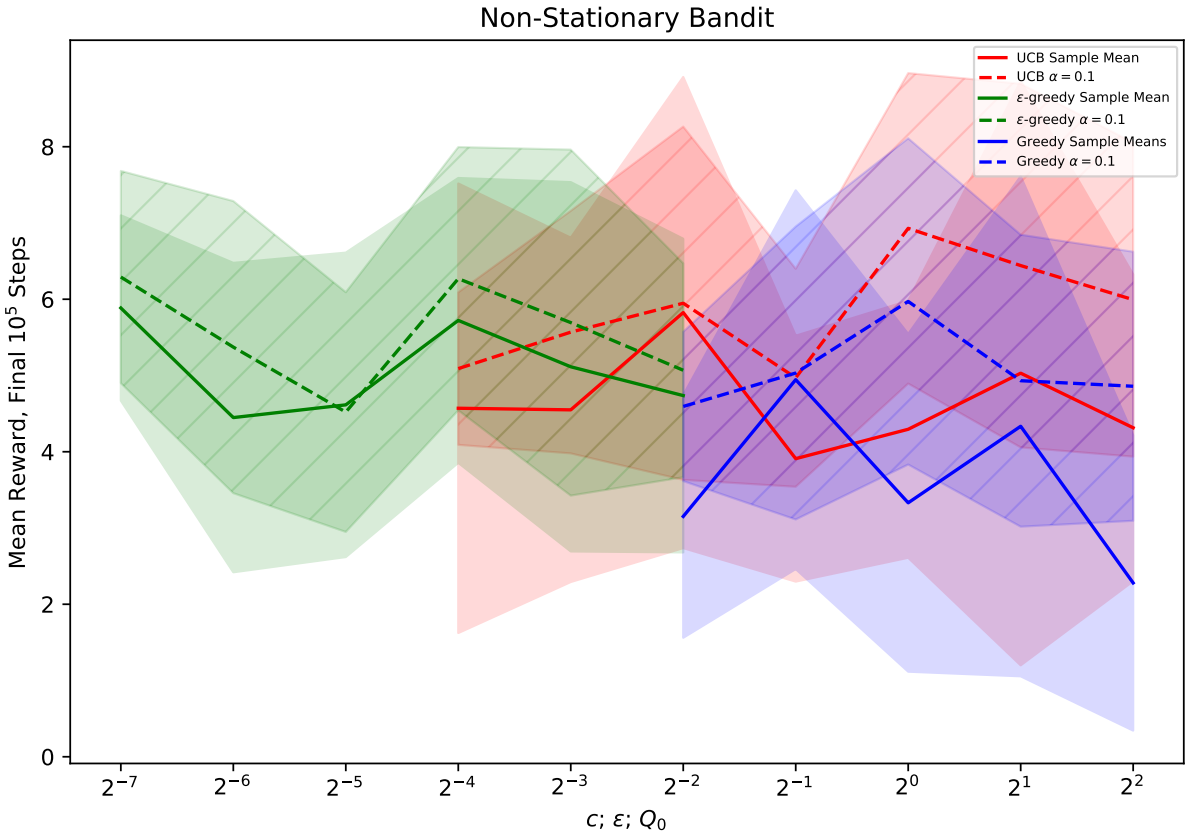
Exercise 2.10: Contextual Bandits

If we do not know which case we are in, we pick the action with the maximum expected reward. However, in this case, it is 0.5 for both actions, marginalised across the cases. The best we can expect to do is to choose arbitarily, receiving average reward 0.5.

Alternatively, if we are informed which case we are in, we can solve this problem by treating it as two seperate bandit problems and applying a typical bandit algorithm, such as UCB.

Exercise 2.11: Non-Stationary Comparison

Please see the below figure.



As expected, the sample average performs worse than the constant step size update. This pattern can consisently be seen.

The graphs are *very* noisy, perhaps because we have only chosen ten seeds. It would have probably been better to fix the bandit problem (or series of bandit problems) - that could itself be a large cause of the noise.

Despire this, the greedy method with an optimistic initialisation does poorly. The exploration is encouraged only at the start, but the method should keep exploring throughout the entire sequence, which will not be happening. It is suprising that UCB does as well as it does especially since the notion of uncertainty is now basically incorrect. It would seem that large values of c, which encourage more exploration, seem to do better using a step size.

If I were to do this again, I would run this with a series of fixed bandit problems i.e., fix the random walk which the bandits go on. This would reduce the variance in the graph and hopefully make it easier to spot patterns.